# CHAINVLA: CHAINING VISION-LANGUAGE-ACTION QUERIES THROUGH A UNIFIED EXECUTION STATE FOR LONG-HORIZON MANIPULATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Humans perform long-horizon manipulation by retaining knowledge of what earlier actions have established while continuously adapting the motion underway. By contrast, action-chunked vision-language-action (VLA) policies repeatedly replan from the current input at each query. Existing methods preserve either long-term task evidence through memory or short-term motion through action reuse and ensembling, leaving the cross-query handoff incomplete. We introduce ChainVLA, a 1.2B-parameter VLA policy that chains successive queries through a single, revisable execution state. Progress Context combines a recurrent Working State with sparse event memory to carry observation-derived task progress, while Motion Tail feeds the preceding prediction's unexecuted continuation into state construction and action generation. Together, the two components condition a decoder that regenerates each action horizon under the latest observation, allowing the carried state to guide the next prediction without fixing it. ChainVLA reaches 62.8% average success on RMBench and 98.8% across four LIBERO suites, while removing Motion Tail or Progress Context reduces RMBench success to 11.2% and 3.0%, respectively. These asymmetric ablations are consistent with motion continuity helping preserve the observation stream from which task progress is inferred.

## 1 INTRODUCTION

Long-horizon manipulation unfolds as a continuous execution in which each decision depends on both the progress established by earlier actions and the movement already underway. When a person places several objects in their designated locations, deciding what to handle next depends on what has already been completed, while completing the current placement requires the ongoing movement to be maintained and revised. These two requirements are supported by prior work on serial behavior (Lashley, 1951; Miller et al., 1960) and sensorimotor control (Wolpert et al., 1995), and they correspond to task progress and motion continuation in Figure 1(b).

In contrast to this continuous organization, action-chunked vision-language-action (VLA) policies execute long tasks through repeated receding-horizon queries, each of which predicts a finite action horizon, executes a strict prefix, and then replans (Zhao et al., 2023; Chi et al., 2025; Black et al., 2025b; Physical Intelligence et al., 2025; Zheng et al., 2026). Although consecutive queries are adjacent in time, each action chunk is generated anew from the current observation, proprioceptive state, and instruction, without an explicit handoff of either the task progress inferred at the preceding query or the unfinished continuation of its prediction. Figure 1(a) shows how this missing handoff creates ambiguity at two coupled levels. At the task level, earlier task evidence may no longer be recoverable from the current view, as illustrated by RMBench's Put Back Block task, in which the same observation and instruction can follow different histories and therefore require different next decisions (Chen et al., 2026a). At the motion level, replanning discards the preceding prediction's unexecuted continuation, so the newly predicted prefix can disagree with the motion it replaces even when the current pose appears similar. Because the motion executed after one query shapes the observation stream available at the next, motion continuity also helps preserve the observations needed to update task progress, suggesting that the two components operate consecutively rather than independently.



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

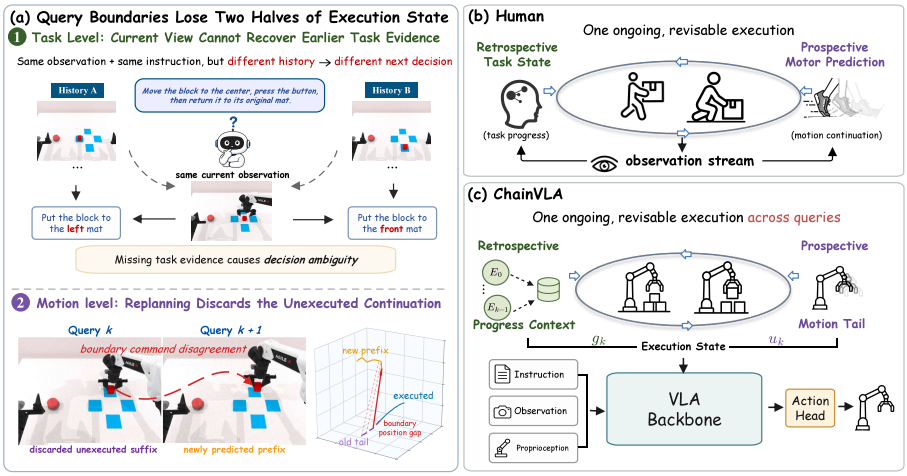



**Figure 1:** ChainVLA motivation and design. (a) Replanning can lose earlier task evidence and unexecuted motion. (b) Humans couple established progress with feedback-revised motion. (c) Progress Context $g_k$ and Motion Tail $u_k$ jointly condition each new horizon.

To recover the cross-query continuity lost by this design, existing attempts follow two complementary lines, with memory-augmented policies retaining recurrent state, selected observations, or compressed task history without preserving the preceding prediction's unexecuted continuation (Shi et al., 2026b; Chen et al., 2026b; Yang et al., 2026; Huang et al., 2026). A second line based on Temporal Ensemble, real-time execution, or native continuation preserves short-term motion by combining or reusing actions, yet it cannot retain task evidence from much earlier in the episode (Zhao et al., 2023; Black et al., 2025a; Liu et al., 2026a). This gap raises a central question: *how can a VLA policy chain task progress and motion continuation across queries while keeping each new prediction responsive to the latest observation?*

To answer this question, we introduce **ChainVLA**, a 1.2B-parameter VLA policy that connects successive queries through a single, revisable execution state (Figure 1(c)). The state comprises *Progress Context*, which carries observation-derived task progress by combining a recurrent Working State with evidence retrieved from a sparse event memory, and *Motion Tail*, which carries the preceding prediction's unexecuted continuation into state construction and action generation. At each query, this state is read, updated, and passed forward, turning separately decoded action chunks into a cross-query chain. Together, the two components condition each new action horizon, while the decoder regenerates every horizon position under the current observation so that the carried state remains a revisable prior rather than a fixed plan (Figure 2). With both components chained across queries, ChainVLA reaches 62.8% average success on RMBench and 98.8% across four LIBERO suites, whereas removing Motion Tail or Progress Context reduces RMBench success to 11.2% and 3.0%, respectively.

Based on this formulation and evaluation, our contributions are

- We identify task progress and unfinished motion as two complementary components of cross-query execution state. Their asymmetric removal effects are consistent with motion continuity preserving the observation stream that informs task-progress inference.
- We introduce ChainVLA, a 1.2B-parameter policy that chains queries through one revisable state of Progress Context and Motion Tail, regenerating each horizon under the latest available observation.
- We validate ChainVLA on RMBench and four LIBERO suites. Matched removals leave both partial states far below the complete model; observation history, temporal ensembling, and post-decoding smoothing also fall short.

## 2 Related Work

**Task progress under partial observability.** Partially observable control represents interaction history through a belief state or another recursively updated statistic (Åström, 1965; Smallwood & Sondik,

1973; Kaelbling et al., 1998), and RMBench and RoboMME make the need explicit for manipulation, where an initial layout, hidden object, or completed subtask can stay relevant after it leaves the current observation (Chen et al., 2026a; Dai et al., 2026). Recent VLA systems preserve this observation-derived evidence through episodic, object-centric, prospective, or multi-scale memory (Chung et al., 2026; Hu et al., 2026; Guo et al., 2026; Torne et al., 2026; Zeng et al., 2026b). MemoryVLA reads and writes a perceptual-cognitive bank at every timestep, then consolidates adjacent entries at capacity (Shi et al., 2026b), whereas sparse alternatives select which observations become long-range records (Chen et al., 2026b; Zeng et al., 2026a; Yang et al., 2026), and recurrent queries and bounded caches propagate local state between calls (Li et al., 2026; Sun et al., 2026a; Lei et al., 2026; Tan et al., 2026). All of these statistics are derived from observations, so none carries what the policy itself last intended.

**Action chunks across policy queries.** Action-chunking policies predict short horizons and repeatedly replan (Zhao et al., 2023; Chi et al., 2025). Temporal ensembling combines separately decoded overlapping horizons at execution time (Zhao et al., 2023), guided test-time sampling improves agreement during decoding (Liu et al., 2024; Malhotra et al., 2025), and real-time flow execution and native continuation reuse overlap during execution or generation (Black et al., 2025a; Liu et al., 2026a). These target local action coordination rather than evidence from much earlier in the episode, so what distinguishes ChainVLA is not prior-conditioned generation alone but combining prediction-derived continuation with observation-derived task evidence in one per-query state update. Long-horizon planners and world models instead decompose subgoals or imagine latent state (Liu et al., 2026b; Shi et al., 2026a; Sun et al., 2026b), while ChainVLA chains calls to one horizon-based policy without requiring a separate planner.

# 3 METHOD

## 3.1 INCOMPLETE STATE HANDOFF

ChainVLA considers history-dependent control in the strict-prefix receding-horizon regime, $1 \leq h_{\text{exec}} < H$. At query $k$, the policy receives an observation $o_k$, proprioceptive state $r_k$, and language instruction $\ell$, grouped as $x_k = (o_k, r_k, \ell)$. It predicts a horizon-$H$ action sequence $A_k = (a_{k,1}, \ldots, a_{k,H})$ and executes the first $h_{\text{exec}}$ actions before querying again (Zhao et al., 2023; Chi et al., 2025). Thus, each new query observes the scene produced by the executed prefix while replacing a nonempty suffix of the preceding prediction.

Let $\mathcal{H}_k$ and $\mathcal{H}'_k$ denote two interaction histories before query $k$, with induced current inputs $x_k$ and $x'_k$. The current input is decision-insufficient when identical inputs admit no shared optimal next executed prefix:

$$x_k = x'_k, \qquad \mathcal{E}^*(\mathcal{H}_k) \cap \mathcal{E}^*(\mathcal{H}'_k) = \varnothing, \tag{1}$$

where $\mathcal{E}^*(\mathcal{H})$ is the set of optimal length-$h_{\text{exec}}$ prefixes. This criterion identifies histories whose difference matters to the next committed control decision. The task-scale case arises when an initial layout, completed subtask, or earlier displacement remains decision-relevant after leaving view (Chen et al., 2026a; Dai et al., 2026; Chen et al., 2026b).

The current input also omits prediction-side context. Because only a prefix of $A_{k-1}$ was executed, its remaining suffix specifies where the preceding query proposed to take the unfinished motion, which is neither completed behavior nor a fixed command, and may need revision once the new observation arrives. A mismatch there extends beyond one boundary: a sharp change in the newly decoded command alters the executed state that contributes to $x_{k+1}$, giving repeated mismatches a route into later queries.

We distinguish the state carried between queries, $m_k = (D_k, W_{k-1}, C_k, u_k)$, from the policy-facing execution context $s_k = (g_k, u_k)$ formed at query $k$. Here $D_k$ records executed actions, $W_{k-1}$ is the preceding Working State, $C_k$ is Sparse Event Memory, and $u_k$ is the preceding prediction's unexecuted suffix. Progress Context $g_k = \text{Fuse}(L_k, Z^e_k)$ is recomputed from the current input and carried state, combining live execution tokens with retrieved event evidence. Motion Tail $u_k$ crosses the boundary directly and remains available to this update and to action generation. The resulting context approximates history-conditioned generation,

$$\Pi_\theta(\cdot \mid x_k, s_k) \approx \Pi^*(\cdot \mid \mathcal{H}_k), \qquad g_k = G_\theta(x_k, m_k), \tag{2}$$

where $\Pi_\theta$ is the learned policy, $\Pi^*$ is an optimal history-conditioned policy over action horizons, and $G_\theta$ comprises the Working State update and event retrieval. Equation 3 specifies how the carried state advances to the next query. The two policy-facing components provide observation-derived task evidence and prediction-derived continuation, respectively, while remaining conditional inputs to a new prediction under $x_k$.

### 3.2 The Execution State

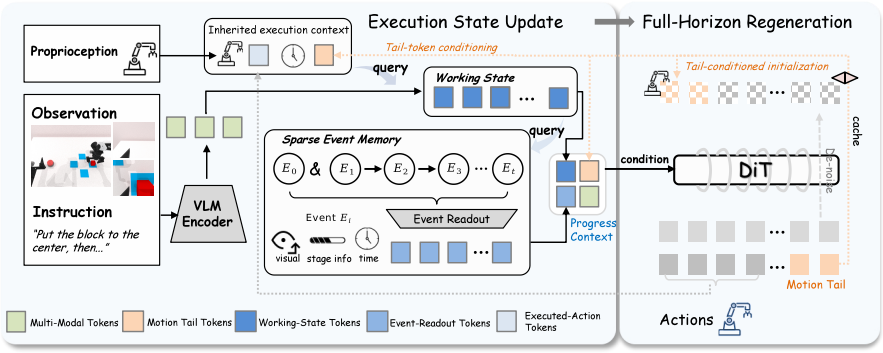



**Figure 2:** ChainVLA query transition. Progress Context combines recurrent Working State and retrieved event evidence. Motion Tail feeds the preceding unexecuted suffix into state construction and generation. The decoder regenerates the horizon under the current observation, prefix execution then advances progress, and the new suffix becomes $u_{k+1}$ for the next query.

ChainVLA turns the incomplete handoff into one read, update, and write transition, while keeping every carried signal revisable under the newest observation. Figure 2 follows this transition from query $k$ to query $k+1$. Query $k$ reads the current input $x_k$ and carried state $m_k$; the eight most recent actions in $D_k$ are encoded as executed-action context. The encoder maps $x_k$ to multimodal tokens $X_k$. Learned working queries then produce live tokens $L_k$ and an execution summary $\bar{W}_k$. The summary retrieves event evidence $Z_k^e$ from $C_k$, and policy-facing fusion forms Progress Context $g_k = \mathrm{Fuse}(L_k, Z_k^e)$. The incoming Motion Tail acts as a continuation prior through two model-side routes, where its tokens condition the Working State and remain in the decoder condition bundle, while its aligned trajectory initializes action generation. The two halves differ in temporal coverage rather than update frequency, since Progress Context can carry evidence across many queries whereas Motion Tail connects only the immediately preceding prediction.

The conditional-flow decoder consumes $X_k$ and samples $A_k \sim \Pi_\theta(\cdot \mid x_k, g_k, u_k)$ with a DiT backbone, so ChainVLA changes the decoder's cross-query context and initialization rather than the action-generator family. The query transition is

$$(A_k, W_k, C_{k+1}) = F_\theta(x_k, D_k, W_{k-1}, C_k, u_k), \qquad u_{k+1} = \mathrm{Tail}(A_k), \tag{3}$$

where $F_\theta$ denotes the query transition and $\mathrm{Tail}(A_k) = (a_{k,h_{\mathrm{exec}}+1}, \ldots, a_{k,H})$ denotes the predicted but unexecuted suffix. This equation describes the transition shared by ordered training and rollout, and during training the carried suffix is detached before it conditions the next query. After prefix execution, the controller appends the executed actions to form $D_{k+1}$ and commits the event candidate only when the dataset-specific write rule fires. Thus $m_{k+1} = (D_{k+1}, W_k, C_{k+1}, u_{k+1})$ is retained for the next query, which recomputes its own Progress Context. Algorithm 1 gives the complete loop; all carried states reset between episodes.

### 3.3 Carrying Task Progress

Progress Context can carry task evidence across multiple queries, including evidence that remains decisive after it leaves the current view. It combines a recurrent Working State for the current query with a retrievable Sparse Event Memory for sparse earlier milestones.

**Recurrent Working State.** The Working State summarizes the current execution and uses that summary to retrieve evidence from earlier queries. Before query $k$, the recurrent state is $W_{k-1} =$

$(L_{k-1}, B_{k-1})$, where $L_{k-1}$ are the previous live tokens and $B_{k-1}$ is a bounded cache of earlier live tokens. The update produces $W_k = (L_k, B_k)$ together with a pooled retrieval summary $\bar{W}_k$. Let $\tau_k$ be the query-time encoding. The executed history and incoming tail are encoded as $P_k = \phi_P(D_k)$ and $U_k = \phi_U(u_k)$. Learned working queries attend to the current multimodal tokens and incorporate the preceding live state:

$$\begin{aligned} \eta_k &= \mathrm{MLP}\left[\phi_r(r_k), \mathrm{Pool}(P_k), \mathrm{Pool}(U_k), \tau_k\right], \\ \widetilde{L}_k &= \mathrm{CrossAttn}(Q_W + \eta_k, X_k), \\ L_k &= \mathrm{LN}\Big(\widetilde{L}_k + m_{k-1}^{\mathrm{live}}\,\mathrm{CrossAttn}(\widetilde{L}_k, L_{k-1})\Big), \\ \bar{W}_k &= \mathrm{Pool}([L_k, B_{k-1}, P_k, U_k]), \end{aligned} \tag{4}$$

where $Q_W$ denotes the learned working-query bank, $\phi_r$ denotes the proprioception encoder, $\eta_k$ is the pooled conditioning vector, $\widetilde{L}_k$ is the current-input update before recurrence, $m_{k-1}^{\mathrm{live}} \in \{0, 1\}$ is the recurrence-validity mask, and LN denotes layer normalization. The mask is zero at episode start and one otherwise. Terms separated by commas inside brackets are concatenated as tokens. Pooled action summaries form the working-query context, while the full encoded sequences contribute to $\bar{W}_k$. The bounded cache admits the most recent live tokens, evicts the oldest slot, and is cleared between episodes. The live tokens $L_k$ enter the policy-facing fusion directly, whereas $\bar{W}_k$ drives event retrieval and, when used, stage estimation. After retrieval, fusion of $L_k$ and $Z_k^e$ forms the policy-facing Progress Context.

**Sparse Event Memory and Retrieval.** Working State updates every query, whereas the Sparse Event Memory changes only on a triggered write. At query $k$, it is the ordered sequence $C_k = \{(\kappa_i, E_i, b_i)\}_{i=1}^{N_k}$, where $N_k$ is the number of stored events. The body $E_i = [V_i, S_i, T_i]$ stores learned visual features, Stage Info, and temporal features, while $b_i$ records the query at which the event was written for age encoding. Learned projections of the current visual features, Working State summary, and query-time encoding produce the candidate $E_k = [V_k, S_k, T_k]$ and its retrieval key $\kappa_k$.

The first valid query is retained as an anchor, after which task-stage transitions determine later writes on stage-annotated data, whereas annotation-free data use a periodic query-time rule. Retrieval is then conditioned on the current execution estimate rather than recency alone. For stage-annotated data, the stage head yields $\pi_k = \mathrm{softmax}(h_s(\bar{W}_k))$ and the current Stage Info summary is $\bar{s}_k = \pi_k$. Without stage annotations, $\bar{s}_k = \mathrm{Pool}(S_k)$. The retrieval query is $q_k^e = \phi_q(\bar{W}_k, \bar{s}_k, \tau_k)$, and each record receives the age-aware score

$$\rho_{k,i} = \cos(\mathrm{LN}(q_k^e), \mathrm{LN}(\kappa_i + \mathrm{Emb}_{\mathrm{age}}(\mathrm{clip}(k - b_i)))), \tag{5}$$

where $\cos(\cdot, \cdot)$ denotes cosine similarity, clip bounds the event age to the supported range, and $\mathrm{Emb}_{\mathrm{age}}$ maps that age to an embedding. The anchor and the highest-scoring non-anchor records are retrieved. Cross-attention fuses their event bodies into readout tokens $Z_k^e$, which are then fused with $L_k$ to form Progress Context, and this read leaves the recurrent state unchanged until its next update.

### 3.4 Carrying Unfinished Motion

Motion Tail carries prediction-derived continuation from the immediately preceding query. After prefix execution, ChainVLA caches the predicted but unexecuted suffix of $A_k$ as $u_{k+1}$. This suffix gives the next query a reference for how the preceding motion was expected to continue. Full-horizon execution leaves no unexecuted suffix but defers observation-conditioned replanning until the horizon ends. Motion Tail instead operates in the strict-prefix regime, carrying the unfinished continuation while retaining frequent replanning. We use *motion intent* for this action-space continuation proposal, not for a separate symbolic plan or latent goal. These actions have not occurred, since $D_k$ records executed behavior, whereas Motion Tail records the continuation proposed by the preceding decoder and imposes no fixed action on the next output. Unlike post-decoding aggregation, Motion Tail is available while the next state and horizon are being formed, before any new output exists.

At query $k + 1$, the incoming tail follows the two model-side routes shown in Figure 2. Action-space normalization and a learned projection encode $u_{k+1}$ as tail tokens $U_{k+1}$ for the Working State. For action generation, $\mathcal{I}$ places the normalized suffix at the start of the new horizon and fills the remaining positions by repeating its last action. The conditional initialization is

$$\mu_{k+1} = \mathcal{I}(u_{k+1}), \qquad \epsilon_{k+1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \qquad \widetilde{A}_{k+1}^{(0)} = \mu_{k+1} + \sigma_u \epsilon_{k+1}, \tag{6}$$

where $\widetilde{A}^{(0)}_{k+1}$ denotes the initial full-horizon decoder state, $\mu_{k+1}$ is its tail-aligned mean, $\epsilon_{k+1}$ is standard Gaussian noise, $\sigma_u > 0$ sets the noise scale, and $\mathbf{I}$ is the identity matrix. Equivalently, $\widetilde{A}^{(0)}_{k+1} \mid u_{k+1} \sim \mathcal{N}(\mu_{k+1}, \sigma_u^2 \mathbf{I})$. The conditional-flow decoder regenerates every horizon position from this state using the latest observation, Progress Context, and tail tokens. Within the decoder, no horizon position is copied or frozen, so the current query may revise any action. After decoding, the controller linearly blends overlapping steps of the regenerated plan and incoming suffix before execution. This post-decoding operation is an execution-side handoff, separate from the two model-side routes. At episode start, the tail path is masked, so the decoder uses its standard noisy initialization without a continuation prior.

### 3.5 Training

Training unrolls the same execution-state transition used at rollout and follows episode query order. During this unroll, each eligible query uses its predecessor's detached predicted suffix with probability 0.5; otherwise, the tail path is masked and the decoder uses its standard noisy initialization. At every step, the model updates the Working State, retrieves Progress Context, predicts an action horizon, conditionally writes an event, and passes its detached Motion Tail forward to directly condition the next query. When task-stage annotations are available, ground-truth transitions supervise the stage estimate and training-time write targets, although the labels never enter the action decoder. At rollout, the predicted stage distribution supplies event semantics and triggers writes, whereas annotation-free domains use the same periodic query-time write rule during both training and rollout phases.

Training combines the standard conditional-flow action objective with stage supervision where annotations are available and an overlap-consistency regularizer for Motion Tail handoff. The regularizer penalizes disagreement over the temporally aligned overlap of consecutive predicted horizons. The preceding prediction is detached, encouraging a coherent handoff without fixing actions in the new horizon. This training objective is distinct from the three-step overlap RMSE used only as a deployment diagnostic. The recurrent and event states follow inference order, and rollout carries only the model-predicted suffix across the boundary.

## 4 Experiments

Our experiments ask three questions. Does chaining execution state across queries support memory-dependent and general manipulation (Section 4.2)? What happens when each component is removed (Section 4.3)? And what is the relationship between motion continuity and memory-dependent task performance (Sections 4.4–4.5)?

### 4.1 Setup

**Benchmarks and comparisons.** We evaluate the five M(1) tasks of RMBench, which require recalling an initial configuration or previously observed information to resolve the current manipulation decision (Chen et al., 2026a). We train one task-adapted model per task, evaluate 100 rollout episodes, and use RMBench as the primary diagnostic setting. LIBERO covers the Spatial, Object, Goal, and Long suites and serves as a breadth check, with one model finetuned per suite and 50 episodes per task (Liu et al., 2023). Comparisons cover action-chunk and pretrained policies together with benchmark and recent memory systems (Chi et al., 2025; Zhao et al., 2023; Black et al., 2025b; Physical Intelligence et al., 2025; Zheng et al., 2026; Kim et al., 2024; Qu et al., 2025; Pertsch et al., 2025; NVIDIA et al., 2025; Li et al., 2025; Chen et al., 2026a; Sridhar et al., 2025; Shi et al., 2026b; Tan et al., 2026; Hu et al., 2026; Liu et al., 2026b).

Table 1 reports baseline results from the cited sources; the internally trained variants provide shared-protocol comparisons of the execution-state design.

**Protocol.** All internally trained conditions share the Florence-2-large backbone (Xiao et al., 2024) and conditional-flow decoder, the same task data with 50 demonstrations per task, 20,000 optimization steps at batch size 4, and the same evaluation protocol, where Full has 1.2B parameters. Queries stay in episode order during training, and every variant receives executed-action history separately from Motion Tail, so a removal withholds carried state rather than access to past actions. The appendix

**Table 1:** Success rates (%) on RMBench and LIBERO. Baseline results are source-reported; ChainVLA uses the protocol above. Parentheses give reported parameter counts, where '+' separates components, and missing counts are not inferred. Bold and underlined values are best and second-best. RMBench averages use the five M(1) tasks.

| RMBench M(1) | | | | | | |
|---|---|---|---|---|---|---|
| Policy | Obs. | Rearr. | Put B. | Swap B. | Swap T. | Avg. |
| DP | 1 | 0 | 0 | 11 | 20 | 6.4 |
| ACT | 1 | 29 | 0 | 2 | 2 | 6.8 |
| $\pi_{0.5}$ (2.6B+0.3B) | <u>9</u> | 13 | 11 | 24 | 15 | 14.4 |
| X-VLA (0.9B) | <u>9</u> | 13 | 18 | 16 | 3 | 11.8 |
| MemER | 7 | 17 | 0 | 14 | 7 | 9.0 |
| MemoryVLA (7B+0.3B) | 0 | 22 | 50 | 17 | 9 | 19.6 |
| Mem-0 (8B+2B) | 4 | 89 | <u>90</u> | <u>67</u> | 14 | <u>52.8</u> |
| MemoAct | 4 | **98** | 41 | – | **55** | – |
| PAM | – | 29 | 30 | 15 | – | – |
| Goal2Skill | 8 | 38 | – | – | – | – |
| **ChainVLA (1.2B)** | **11** | <u>93</u> | **96** | **74** | <u>40</u> | **62.8** |

| LIBERO | | | | | |
|---|---|---|---|---|---|
| Policy | Spatial | Object | Goal | Long | Avg. |
| OpenVLA (7B) | 84.7 | 88.4 | 79.2 | 53.7 | 76.5 |
| SpatialVLA (3.5B) | 88.2 | 89.9 | 78.6 | 55.5 | 78.1 |
| $\pi_0$-FAST (3B) | 96.4 | 96.8 | 88.6 | 60.2 | 85.5 |
| GR00T-N1 (2.2B) | 94.4 | 97.6 | 93.0 | 90.6 | 93.9 |
| $\pi_0$ (3.3B) | 96.8 | 98.8 | 95.8 | 85.2 | 94.2 |
| $\pi_{0.5}$ + KI | 98.0 | 97.8 | 95.6 | 85.8 | 94.3 |
| PAM | 86.1 | 97.0 | 94.5 | 84.7 | 90.6 |
| MemoryVLA (7B+0.3B) | **98.4** | 98.4 | 96.4 | 93.4 | 96.7 |
| CronusVLA (7B) | 97.3 | **99.6** | 96.9 | 94.0 | 97.0 |
| X-VLA (0.9B) | <u>98.2</u> | 98.6 | <u>97.8</u> | <u>97.6</u> | <u>98.1</u> |
| **ChainVLA (1.2B)** | **98.4** | <u>99.4</u> | **99.2** | **98.2** | **98.8** |

specifies each intervention, the boundary-metric definitions, and the remaining training and rollout hyperparameters.

**Metrics.** The primary metric is task success rate over 100 episodes per RMBench task and 50 per LIBERO task. RMBench macro averages include only methods with complete five-task coverage, and LIBERO uses an unweighted four-suite average while retaining Long separately.

To inspect what happens at a query boundary, we adapt four lower-is-better diagnostics from prior work on action-chunk transitions (Black et al., 2025a; Liu et al., 2026a; Zhan et al., 2026), where position and orientation command discontinuity ($\mathrm{CD}_p$, $\mathrm{CD}_R$) measure the pose gap at replanning, boundary second difference (B2) the command-velocity change across it, and three-step overlap RMSE ($\mathrm{RMSE}_3$) the disagreement between the preceding plan's unexecuted suffix and the next plan. The first three read executed commands and the last compares plans, and none measures physical smoothness or accumulated state deviation. We pair Full and w/o Motion Tail rollouts under matched scenes on Put Back Block (20 pairs), Rearrange Blocks (30), and Swap Blocks (30), and pool task means by pair count. The appendix gives the formulas and all per-task values for these matched rollout conditions.

### 4.2 Main Results

On the five M(1) tasks of RMBench, ChainVLA reaches 62.8 average success at 1.2B parameters, ahead of the strongest listed method with complete coverage (Mem-0, 52.8, at 8B+2B), and it is best or second-best on every task (Table 1). Methods without complete coverage are excluded from the average. The gap is largest where a subtask must be repeated or reversed: Put Back Block requires returning a block to a pad that is no longer identifiable in the current view, and ChainVLA reaches 96 against 90 for Mem-0 and 50 for MemoryVLA. On LIBERO the policy averages 98.8 with every

suite above 98, including 98.2 on LIBERO-Long. LIBERO is near ceiling for recent policies, so it establishes breadth rather than separating designs, and the analysis below therefore uses RMBench, where all conditions are trained and evaluated under the same matched protocol.

**Table 2:** RMBench ablations (100 episodes per task). Best non-Full values are underlined. "w/o Stage Ann." uses periodic three-query writes without stage supervision. "w/o Both" retains current observations and executed actions. "FIFO Hist. + TE" substitutes fixed-length observation history and post-decoding temporal ensembling for carried state.

| Configuration | Obs. | Rearr. | Put B. | Swap B. | Swap T. | Avg. |
|---|---|---|---|---|---|---|
| *Task-progress side* (motion continuation kept) | | | | | | |
| w/o Stage Ann. | <u>10</u> | <u>84</u> | <u>78</u> | <u>61</u> | 34 | <u>53.4</u> |
| w/o Live Tokens | <u>10</u> | 78 | 72 | 48 | 28 | 47.2 |
| w/o Event Readout | 9 | 65 | 25 | 38 | 24 | 32.2 |
| w/o Progress Ctx. | 6 | 4 | 0 | 1 | 4 | 3.0 |
| *Motion-continuation side* (task progress kept) | | | | | | |
| w/o Tail Tokens | 8 | 78 | <u>78</u> | 58 | <u>38</u> | 52.0 |
| w/o Traj. Init. | 7 | 60 | 45 | 45 | 33 | 38.0 |
| w/o Motion Tail | 1 | 24 | 0 | 2 | 29 | 11.2 |
| *No carried state, and a substitute* | | | | | | |
| w/o Both | 4 | 0 | 0 | 0 | 4 | 1.6 |
| FIFO Hist. + TE | 8 | 58 | 47 | 40 | 25 | 35.6 |
| **ChainVLA (Full)** | **11** | **93** | **96** | **74** | **40** | **62.8** |

### 4.3 Ablation Analysis

**Pathway removals.** Table 2 evaluates the two carried pathways under the shared training and rollout protocol. Retaining task progress without Motion Tail gives 11.2 average success, retaining Motion Tail without Progress Context gives 3.0, and removing both gives 1.6, compared with 62.8 for Full. Whole-pathway removals assess system dependence by withdrawing the inputs, training terms, and execution handling together; targeted variants examine the routes within each pathway.

**Within-pathway variants.** Removing live tokens gives 47.2, while removing event readout gives 32.2; on Put Back Block, the latter reduces success from 96 to 25, consistent with the task's need to recall an earlier placement. On the motion side, retaining tail-aligned initialization without tail tokens gives 52.0, while retaining tail tokens without that initialization gives 38.0. Both routes contribute under the evaluated protocol, with initialization producing the larger removal effect.

**Annotation-free writing and history substitution.** Periodic writing without stage supervision reaches 53.4, demonstrating operation without task-stage labels. Full reaches 62.8 with stage-aware writing; the comparison varies both stage supervision and the write rule. FIFO History + Temporal Ensemble supplies observation history before decoding and combines overlapping predictions afterward. It reaches 35.6, below Full on every task and 27.2 points below overall, supporting the benefit of the carried execution state over this tested substitute.

### 4.4 Motion Continuity and Memory

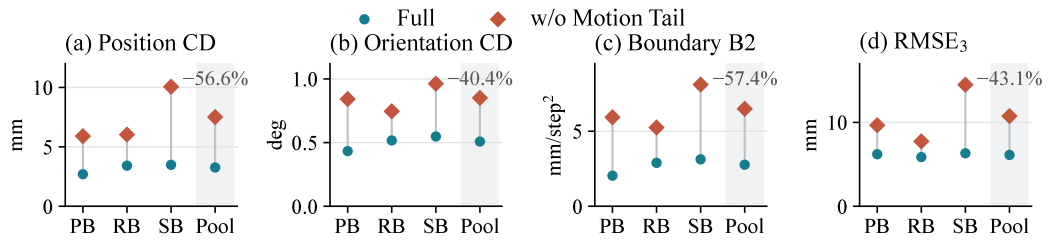



**Figure 3:** Boundary errors for Full and w/o Motion Tail. PB/RB/SB denote Put Back, Rearrange, and Swap Blocks, and Pool weights their 20/30/30 rollout pairs. Lower is better. Panels (a–c) use executed commands and (d) final plans.

RMBench is a memory benchmark, so the size of the motion-side effect needs explaining: removing motion continuation leaves the memory pathway fully intact, yet success falls from 62.8 to 11.2. The likely reason is that the two halves act as consecutive links rather than parallel abilities. When each horizon is sampled without reference to the one it replaces, consecutive commands can disagree at the boundary: on one matched Put Back Block boundary, the position gap is 17.32 mm without Motion Tail against 0.66 mm with it (Appendix Figure A2). Across all 80 matched pairs, Full lowers every measure, with pooled reductions of 56.6%, 40.4%, 57.4%, and 43.1% for $\mathrm{CD}_p$, $\mathrm{CD}_R$, B2, and $\mathrm{RMSE}_3$ (Figure 3). A displacement of this size repeated at every boundary moves the arm along poses the demonstrations never visit, and recovering from them costs additional replanning queries.

That recovery is what reaches the memory pathway. Progress Context is built from observations, so the sequence of viewpoints the arm visits is its input. An episode spent recovering visits different viewpoints, in a different order, than the demonstrations that trained the retrieval, so evidence needed later may never be written or may be retrieved against an off-distribution query. Motion continuity therefore appears to support reliable task-progress estimation rather than to act as a competing objective, which is also consistent with the asymmetry in Table 2. Removing task progress leaves a policy that moves coherently toward the wrong subgoal and fails cleanly at 3.0, whereas removing motion continuation degrades the very input the surviving memory pathway reads. Because the paired policies diverge within matched scenes, these boundary measurements establish an association rather than component-level causation.

### 4.5 Post-Decoding Smoothing Controls

**Table 3:** Post-decoding controls. Success uses 100 episodes per task, while position CD (mm, lower is better) uses 20 Put Back and 30 Swap pairs. The appendix reports the other boundary measures.

| | Put Back Block | | Swap Blocks | |
|---|---|---|---|---|
| Method | Succ. | $\mathrm{CD}_p$ | Succ. | $\mathrm{CD}_p$ |
| w/o Motion Tail | 0 | 5.90 | 2 | 10.06 |
| Linear Cont. | 0 | **2.09** | 2 | 3.72 |
| Temporal Ens. | 0 | 3.57 | 0 | 7.52 |
| **Full** | **96** | 2.69 | **74** | **3.49** |

If the gain came from smoother boundaries, then producing smooth boundaries by any means should recover it. It does not. Linear Continuation reaches boundary values below Full on all four Put Back Block measures and three of four on Swap Blocks, yet succeeds in 0% and 2% of episodes against 96% and 74% for Full (Table 3). Temporal Ensemble also reduces every measure relative to w/o Motion Tail and succeeds in 0% of episodes on both tasks.

The comparison isolates when the suffix is used rather than whether it is used at all. Full also blends overlapping steps after decoding, and Linear Continuation is exactly Full with that blend retained and the model-side conditioning removed, so the blend is held fixed and 96% against 0% is attributable to the pathway that differs. Smoothing acts on already-decoded commands, so it can only interpolate between a plan and the motion it replaces, and the plan is fixed by the time it applies. Carrying the suffix into state construction and generation instead changes what is decoded. The boundary metrics are therefore diagnostics, not objectives: Full is not the smoothest condition here, yet the smoothest conditions still fail to recover task success.

## 5 Conclusion

ChainVLA carries observation-derived task progress and the preceding prediction's revisable suffix across otherwise independent receding-horizon queries. In our fixed-rate evaluation, neither partial state approaches the complete policy on RMBench, and their asymmetry is consistent with motion continuity preserving task-progress observations.

## AI USE STATEMENT

Generative AI tools assisted with drafting and revising text, identifying and checking relevant literature, refining the presentation of the method and experimental methodology, and analyzing and interpreting reported results. They also assisted with figure preparation and LaTeX formatting. AI-assisted revisions were checked against the source manuscript, cited references, and available experimental records. The authors take responsibility for the final content of this work, including its text, claims, and artifacts produced with the aid of generative AI.

## REFERENCES

Karl Johan Åström. Optimal control of markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10:174–205, 1965. doi: 10.1016/0022-247X(65)90154-X.

Kevin Black, Manuel Y. Galliker, and Sergey Levine. Real-time execution of action chunking flow policies, 2025a. URL https://arxiv.org/abs/2506.07339.

Kevin Black et al. $\pi_0$: A vision-language-action flow model for general robot control. In *Proceedings of Robotics: Science and Systems*, 2025b.

Tianxing Chen, Yuran Wang, Mingleyang Li, Yan Qin, Hao Shi, Zixuan Li, Yifan Hu, Yingsheng Zhang, Kaixuan Wang, Yue Chen, et al. RMBench: Memory-dependent robotic manipulation benchmark with insights into policy design, 2026a.

Yipeng Chen et al. Non-markovian long-horizon robot manipulation via keyframe chaining, 2026b.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10–11):1684–1704, 2025. doi: 10.1177/02783649241273668.

Nhat Chung et al. Rethinking progression of memory state in robotic manipulation: An object-centric perspective. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 3407–3415, 2026. doi: 10.1609/aaai.v40i5.37337.

Yinpei Dai, Hongze Fu, Jayjun Lee, Yuejiang Liu, Haoran Zhang, Jianing Yang, Chelsea Finn, Nima Fazeli, and Joyce Chai. RoboMME: Benchmarking and understanding memory for robotic generalist policies. In *Proceedings of the International Conference on Machine Learning*, 2026. Oral presentation.

Xinying Guo et al. Chameleon: Control-indexed prospective memory for visuomotor manipulation, 2026.

Qingda Hu, Ziheng Qiu, Zijun Xu, Kaizhao Zhang, Xizhou Bu, Zuolei Sun, Bo Zhang, Jieru Zhao, Zhongxue Gan, and Wenchao Ding. Resolving state ambiguity in robot manipulation via adaptive working memory recoding. *IEEE Robotics and Automation Letters*, 11(7):8439–8446, 2026. doi: 10.1109/LRA.2026.3693565.

Yuzhi Huang, Jie Wu, Weijue Bu, Ziyi Xiong, Gaoyang Jiang, Ye Li, Kangye Ji, Shuzhao Xie, Yue Huang, Chenglei Wu, Jingyan Jiang, and Zhi Wang. RoboStream: Weaving spatio-temporal reasoning with memory in vision-language models for robotics, 2026.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998. doi: 10.1016/S0004-3702(98)00023-X.

Moo Jin Kim et al. OpenVLA: An open-source vision-language-action model. In *8th Annual Conference on Robot Learning*, 2024.

Karl S. Lashley. The problem of serial order in behavior. In Lloyd A. Jeffress (ed.), *Cerebral Mechanisms in Behavior: The Hixon Symposium*, pp. 112–136. Wiley, New York, 1951.

Yuheng Lei et al. VPWEM: Non-markovian visuomotor policy with working and episodic memory, 2026.

Hang Li et al. ReMem-VLA: Empowering vision-language-action model with memory via dual-level recurrent queries, 2026.

Hao Li, Shuai Yang, Yilun Chen, Xinyi Chen, Xiaoda Yang, Yang Tian, Hanqing Wang, Tai Wang, Dahua Lin, Feng Zhao, and Jiangmiao Pang. CronusVLA: Towards efficient and robust manipulation via multi-frame vision-language-action modeling, 2025.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, 2023.

Yuejiang Liu, Jubayer Ibn Hamid, Annie Xie, Yoonho Lee, Maximilian Du, and Chelsea Finn. Bidirectional decoding: Improving action chunking via guided test-time sampling, 2024. URL https://arxiv.org/abs/2408.17355.

Yufeng Liu, Hang Yu, Juntu Zhao, Bocheng Li, Di Zhang, Mingzhu Li, Wenxuan Wu, Yingdong Hu, Junyuan Xie, Junliang Guo, Dequan Wang, and Yang Gao. Learning native continuation for action chunking flow policies, 2026a. URL https://arxiv.org/abs/2602.12978.

Zhen Liu et al. Goal2Skill: Long-horizon manipulation with adaptive planning and reflection, 2026b.

Rhea Malhotra, Yuejiang Liu, and Chelsea Finn. Self-guided action diffusion, 2025. URL https://arxiv.org/abs/2508.12189.

George A. Miller, Eugene Galanter, and Karl H. Pribram. *Plans and the Structure of Behavior*. Holt, Rinehart and Winston, New York, 1960.

NVIDIA, Johan Bjorck, Fernando Castaneda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, et al. GR00T N1: An open foundation model for generalist humanoid robots, 2025.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient action tokenization for vision-language-action models, 2025.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. In *9th Annual Conference on Robot Learning*, 2025. Oral presentation.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, and Xuelong Li. SpatialVLA: Exploring spatial representations for visual-language-action model, 2025.

Hao Shi, Weiye Li, Bin Xie, Yulin Wang, Renping Zhou, Tiancai Wang, Xiangyu Zhang, Ping Luo, and Gao Huang. MemoryVLA++: Temporal modeling via memory and imagination in vision-language-action models, 2026a. URL https://arxiv.org/abs/2606.09827.

Hao Shi et al. MemoryVLA: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. In *The Fourteenth International Conference on Learning Representations*, 2026b.

Richard D. Smallwood and Edward J. Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations Research*, 21(5):1071–1088, 1973. doi: 10.1287/opre.21.5.1071.

Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. MemER: Scaling up memory for robot control via experience retrieval, 2025.

Jun Sun et al. TempoFit: Plug-and-play layer-wise temporal KV memory for long-horizon vision-language-action manipulation, 2026a.

Xiaoquan Sun, Ruijian Zhang, Chen Cao, Yihan Sun, Jiahui Chen, Zetian Xu, Bo Chen, Haijier Chen, Zhen Yang, Jiarun Zhu, Yijun Hong, JingZhe Xu, Jingrui Pang, Mingqi Yuan, and Jiayu Chen. HiMem-WAM: Hierarchical memory-gated world action models for robotic manipulation, 2026b. URL https://arxiv.org/abs/2606.10363.

Liufan Tan, Jiale Li, and Gangshan Jing. MemoAct: Atkinson–shiffrin-inspired memory-augmented visuomotor policy for robotic manipulation, 2026.

Marcel Torne, Karl Pertsch, Homer Walke, Kyle Vedder, Suraj Nair, Brian Ichter, Allen Z. Ren, Haohuan Wang, Jiaming Tang, Kyle Stachowicz, Karan Dhabalia, Michael Equi, Quan Vuong, Jost Tobias Springenberg, Sergey Levine, Chelsea Finn, and Danny Driess. MEM: Multi-scale embodied memory for vision language action models, 2026. URL https://arxiv.org/abs/2603.03596.

Daniel M. Wolpert, Zoubin Ghahramani, and Michael I. Jordan. An internal model for sensorimotor integration. *Science*, 269(5232):1880–1882, 1995.

Bin Xiao, Haiping Wu, Weijian Xu, Xiyang Dai, Houdong Hu, Yumao Lu, Michael Zeng, Ce Liu, and Lu Yuan. Florence-2: Advancing a unified representation for a variety of vision tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4818–4829, 2024. doi: 10.1109/CVPR52733.2024.00461.

Ganlin Yang, Zhangzheng Tu, Yuqiang Yang, Sitong Mao, Junyi Dong, Tianxing Chen, Jiaqi Peng, Jing Xiong, Jiafei Cao, Jifeng Dai, Wengang Zhou, Yao Mu, and Tai Wang. EventVLA: Event-driven visual evidence memory for long-horizon vision-language-action policies, 2026.

Yihan Zeng, Minghao Ye, Yiyuan Chen, Yide Shentu, Philipp Wu, Zike Yan, and Zhongyu Li. KEMO: Event-driven keyframe memory for long-horizon robot manipulation with VLA policies, 2026a.

Zijian Zeng, Fei Ding, Huiming Yang, and Xianwei Li. HELM: Harness-enhanced long-horizon memory for vision-language-action manipulation, 2026b. URL https://arxiv.org/abs/2604.18791.

Dijia Zhan, Xuemiao Xu, Jinyi Li, and Jie Tang. SEAM: Smooth execution of action-chunked motion for vision-language-action policies, 2026.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proceedings of Robotics: Science and Systems XIX*, 2023. doi: 10.15607/RSS.2023.XIX.016.

Jinliang Zheng et al. X-VLA: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *The Fourteenth International Conference on Learning Representations*, 2026.

# A Query Transition and Rollout Details

At query $k$, ChainVLA reads the current multimodal input $x_k$ and persistent state $m_k = (D_k, W_{k-1}, C_k, u_k)$: executed-action history, preceding Working State, Sparse Event Memory, and incoming Motion Tail. Progress Context $g_k$ is derived anew at each query, while $m_k$ contains the quantities retained between queries. It updates the Working State, retrieves task-relevant event evidence, and conditions a full-horizon prediction on Progress Context and Motion Tail. After executing a prefix, the controller appends the executed actions to form $D_{k+1}$, may commit an event according to the dataset-specific write rule, and carries the predicted but unexecuted suffix forward as $u_{k+1}$:

$$\begin{aligned}(A_k, W_k, C_{k+1}) &= F_\theta(x_k, D_k, W_{k-1}, C_k, u_k),\\ u_{k+1} &= \mathrm{Tail}(A_k),\end{aligned} \tag{A1}$$

where $F_\theta$ denotes the query transition and $\mathrm{Tail}(A_k) = (a_{k,h_{\mathrm{exec}}+1}, \ldots, a_{k,H})$ is the unexecuted suffix. This transition is shared by ordered training and rollout, with the carried suffix detached before it conditions the next query during training. All carried states and validity masks reset between episodes, and Algorithm 1 states the induced loop.

**Table A1:** Cross-query execution-state components at a glance.

| Component | Contents | Persistence |
|---|---|---|
| Live tokens | 8 current plus 24 prior tokens in a three-slot FIFO | Recurrently updated every query |
| Executed-action tokens $P_k$ | 4 tokens encoded from the 8 most recent actions in $D_k$ | Re-encoded from $D_k$ at each query |
| Tail tokens $U_k$ | 4 tokens encoded from the incoming Motion Tail $u_k$ | Re-encoded from $u_k$ at each query; zeroed at episode start |
| Event record | 64 visual + 8 Stage Info + 8 temporal tokens, retrieval key, write-query index | Capacity 16, oldest non-anchor evicted |

## A.1 Working State

Table A1 summarizes the cross-query execution-state components. The Working State $W_k = (L_k, B_k)$ contains the current live tokens and their bounded FIFO cache, while executed-action tokens and tail tokens are constructed separately from $D_k$ and $u_k$. Eight learned queries attend to the current multimodal tokens and recurrently update the live tokens, which enter Progress Context directly. A pooled summary of the live, FIFO, executed-action, and tail tokens contributes to the event-retrieval query.

## A.2 Sparse Event Memory and Stage Info

Each stored event comprises 64 visual, 8 Stage Info, and 8 temporal tokens, together with a retrieval key and write-query index. Executed-action and tail tokens contribute to event construction through the retrieval summary $\bar{W}_k$ but are not stored as separate fields in the event record. The first valid event is retained as an anchor. On RMBench, stage transitions supervise training writes, while rollout writes require a predicted stage change with confidence at least 0.55; stage labels do not enter the action decoder. Retrieval uses 32 clipped age buckets to read the anchor and the two highest-scoring non-anchor records. Cross-attention fuses their event bodies into readout tokens without overwriting the Working State.

## A.3 Motion Tail and Ordered Training

Motion Tail follows two model-side routes: the incoming suffix is encoded into tail tokens that condition Working State construction (Table A1), while an aligned action-space representation initializes the full horizon with additive noise. Alignment places the $H - h_{\mathrm{exec}}$ suffix actions at the start of the next horizon and repeats the suffix endpoint thereafter, in both ordered training and rollout. For the 30/24 protocol, six positions carry the suffix and the remaining 24 use its endpoint before adding initialization noise. Motion intent denotes this continuation proposal rather than a symbolic plan or latent goal. The decoder uses both routes but regenerates every position under the current observation and Progress Context. After decoding, the controller linearly blends overlapping steps of

**Algorithm 1** ChainVLA rollout for one episode

$W_{-1} \leftarrow \varnothing, C_0 \leftarrow \varnothing, D_0 \leftarrow \varnothing, u_0 \leftarrow \varnothing$
$k \leftarrow 0$
**while** episode not terminated **do**
  observe $x_k = (o_k, r_k, \ell)$ and encode to $X_k$
  $W_k, \bar{W}_k \leftarrow \textsc{UpdateWorking}(X_k, W_{k-1}, D_k, u_k)$
  $\bar{s}_k \leftarrow \textsc{StageSummary}(\bar{W}_k)$
  $Z_k^e \leftarrow \textsc{Retrieve}(C_k, \bar{W}_k, \bar{s}_k, \tau_k)$
  $g_k \leftarrow \mathrm{Fuse}(L_k, Z_k^e)$
  $A_k \sim \Pi_\theta(\cdot \mid x_k, g_k, u_k)$ ▷ every position regenerated
  $A_k \leftarrow \textsc{BlendOverlap}(A_k, u_k)$ ▷ execution-side handoff
  execute prefix $(a_{k,1}, \ldots, a_{k,h_{\mathrm{exec}}})$
  $D_{k+1} \leftarrow \textsc{Append}(D_k, \text{executed prefix})$
  $C_{k+1} \leftarrow \textsc{MaybeWriteEvent}(C_k, W_k)$
  $u_{k+1} \leftarrow \mathrm{Tail}(A_k)$ ▷ carried as a revisable prior
  $k \leftarrow k + 1$
**end while**

the regenerated plan and incoming suffix before execution. This execution-side handoff is separate from the two model-side routes, and the first query uses standard initialization instead.

Training processes queries in episode order. During this unroll, each eligible query uses its predecessor's detached predicted tail with probability 0.5; otherwise, the tail path is masked and standard noisy initialization is used. The overlap-consistency loss has weight 0.2 when Motion Tail is enabled and is distinct from deployment overlap RMSE.

## B Evaluation and Intervention Details

**Common training and rollout protocol.** All internally trained conditions share the protocol in Table A2 and the same evaluation protocol. Every boundary leaves a six-step unexecuted suffix as Motion Tail. Queries remain ordered, and every variant receives executed-action history separately from Motion Tail, so a removal withholds carried state rather than access to past actions. RMBench stage labels supervise training where enabled and rollouts use predicted stages, while w/o Stage Annotation and annotation-free LIBERO write every three queries during training and rollout.

**Table A2:** Shared training and rollout protocol for all internally trained conditions.

| Item | Setting |
|---|---|
| Vision-language backbone | Florence-2-large |
| Action decoder | Conditional flow |
| Demonstrations per task / steps | 50 / 20,000 |
| Batch size | 4 |
| Horizon $H$ / executed prefix $h_{\mathrm{exec}}$ | 30 / 24 |
| Full parameters | 1.2B |
| Learning rates | Backbone $1 \times 10^{-5}$, decoder and memory $1 \times 10^{-4}$ |
| Learning-rate schedule | Cosine annealing |
| Training hardware | 4× NVIDIA L20 GPUs |
| Random seed / framework | 42 (single seed) / PyTorch under Linux |

**RMBench.** The five evaluated RMBench M(1) tasks are bimanual and require several placements in sequence, so a single episode passes through multiple stages and therefore multiple query boundaries. Each internally evaluated condition uses 100 rollout episodes per task. Figure A1 illustrates the intended terminal configuration for each task.

**LIBERO.** LIBERO serves as a breadth check over the Spatial, Object, Goal, and Long suites, with one model finetuned per suite. We evaluate all 10 tasks in each suite with 50 rollout episodes per task. The suites carry no stage labels, so writes follow the annotation-free schedule and the event chain grows on the fixed three-query interval in both training and rollout.

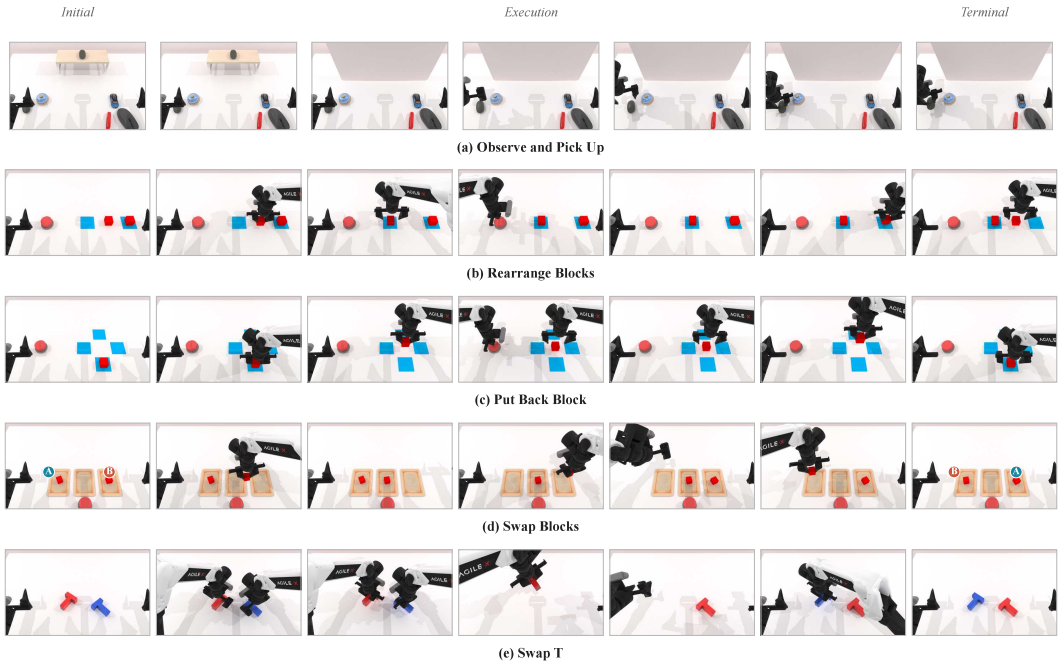



**Figure A1:** RMBench task overview. Each row shows seven snapshots from one successful trajectory, ordered from the initial scene to the terminal state.

**Optimization and infrastructure.** Every condition trains from a single random seed with the learning rates, schedule, hardware, and framework listed in Table A2.

**Matched interventions.** Matched controls cover all five RMBench tasks under the evaluation protocol above. w/o Live Tokens retains the Working State for retrieval, w/o Event Readout removes retrieved evidence, and the Motion Tail controls remove either its Working State tokens or its trajectory initialization. w/o Progress Context also removes its stage objective. Table A3 gives the complete intervention semantics.

**Table A3:** Matched interventions relative to Full. TE denotes Temporal Ensemble.

| Configuration | Change from Full |
|---|---|
| w/o Stage Annotation | Writes every three queries without stage supervision; Motion Tail retained |
| w/o Live Tokens | No live tokens in Progress Context; Working State still supports retrieval |
| w/o Event Readout | No retrieved event readout; recurrent live tokens retained |
| w/o Progress Context | No live tokens, event readout, or stage supervision; Motion Tail retained |
| w/o Tail Tokens | No tail tokens in Working State; tail-aligned initialization retained |
| w/o Trajectory Init. | Standard rather than tail-aligned initialization; tail tokens retained |
| w/o Motion Tail | No model-side tail routes, overlap training, or overlap blending; Progress Context retained |
| w/o Both | No Progress Context or Motion Tail; current observation, action history, and standard initialization retained |
| FIFO History + TE | FIFO observations, standard initialization, and Temporal Ensemble replace both carried contexts |

**Boundary rollouts and output-level controls.** We pair Full and w/o Motion Tail rollouts under the same scenes on Put Back Block (20 pairs), Rearrange Blocks (30), and Swap Blocks (30), pooling task means by pair count. On Put Back Block and Swap Blocks, Linear Continuation retains only the execution-side overlap blend without Motion Tail's model-side conditioning, while Temporal Ensemble aggregates decoded actions over time. Their success rates use 100 episodes per task and

method, and boundary means use the paired rollouts. For compactness, the boundary tables use *no-tail* for w/o Motion Tail.

# C Boundary Diagnostics and Results

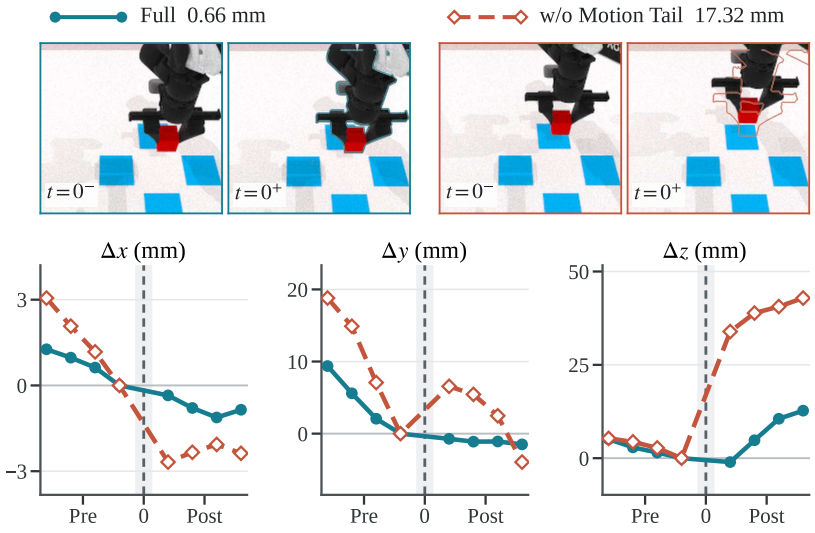



**Figure A2:** Illustrative Put Back Block boundary from paired-scene rollouts. Frames overlay the pre-boundary arm contour, and curves show right end-effector offsets from the $t = 0$ anchor. This single case has 0.66/17.32 mm position CD for Full/w/o Motion Tail. Figure 3 gives aggregate results.

## C.1 Diagnostic Definitions

For a valid boundary $b$ and arm $a \in \{1, 2\}$, let $(p^-_{b,a}, R^-_{b,a})$ be the final executed Cartesian pose command before replanning, and let $(p^+_{b,j,a}, R^+_{b,j,a})$ denote the command at post-boundary index $j$. Averaging over the two arms, position CD, orientation CD, and translational boundary second difference are

$$\begin{aligned} \mathrm{CD}_p(b) &= \frac{1}{2}\sum_{a=1}^{2} \|p^+_{b,0,a} - p^-_{b,a}\|_2, \\ \mathrm{CD}_R(b) &= \frac{1}{2}\sum_{a=1}^{2} \mathrm{angle}\Big((R^-_{b,a})^\top R^+_{b,0,a}\Big), \\ \mathrm{B2}(b) &= \frac{1}{2}\sum_{a=1}^{2} \Big\|(p^+_{b,1,a} - p^+_{b,0,a}) - (p^+_{b,0,a} - p^-_{b,a})\Big\|_2, \end{aligned} \tag{A2}$$

where $\mathrm{angle}(R)$ denotes the $\mathrm{SO}(3)$ geodesic angle and the two finite differences in B2 denote consecutive command increments. We report $\mathrm{CD}_p$ and B2 in millimeters and $\mathrm{CD}_R$ in degrees. All three use the executed high-level command stream.

A boundary counts as valid when the preceding query contributed at least one executed command and the following query contributed at least two, since B2 needs two consecutive post-boundary increments. The first query of an episode has no predecessor and therefore contributes no boundary, and an episode that terminates immediately after replanning contributes none at that seam. Both configurations are scored by the same rule on the same scenes, so the two arms of each pair see the same set of admissible seams up to where their rollouts diverge.

For consecutive final plans, let $p^{\text{old}}_{b,j,a}$ and $p^{\text{new}}_{b,j,a}$ denote the preceding suffix and new-plan positions over three aligned steps:

$$\text{RMSE}_3(b) = \sqrt{\frac{1}{6}\sum_{j=0}^{2}\sum_{a=1}^{2}\left\|p^{\text{old}}_{b,j,a} - p^{\text{new}}_{b,j,a}\right\|_2^2}, \tag{A3}$$

where $j \in \{0,1,2\}$ and $a \in \{1,2\}$ index the aligned steps and arms. Boundary values are averaged within each rollout and then across rollouts within each task. Let $c$ denote Full or no-tail (w/o Motion Tail). For metric $m$, the pooled mean and relative reduction are

$$\begin{aligned} \bar{m}^{c}_{\text{pool}} &= \frac{20\bar{m}^{c}_{\text{Put}} + 30\bar{m}^{c}_{\text{Rearr.}} + 30\bar{m}^{c}_{\text{Swap}}}{80}, \\ \Delta_{\text{rel}} &= 100\left(1 - \frac{\bar{m}^{\text{Full}}_{\text{pool}}}{\bar{m}^{\text{no-tail}}_{\text{pool}}}\right)\%, \end{aligned} \tag{A4}$$

where $\bar{m}^{c}_{t}$ denotes the rollout-averaged task mean and $\Delta_{\text{rel}}$ denotes the pooled relative reduction from no-tail to Full.

### C.2 Full versus w/o Motion Tail

**Table A4:** Full/no-tail boundary results over 20, 30, and 30 rollout pairs. Position CD, boundary second difference, and overlap RMSE use millimeters. Orientation CD uses degrees. Pooled means are count-weighted. Lower is better.

| Task | Config. | $\text{CD}_p$ | $\text{CD}_R$ | B2 | $\text{RMSE}_3$ |
|---|---|---|---|---|---|
| Put Back | no-tail | 5.900 | 0.842 | 5.930 | 9.660 |
| | **Full** | 2.694 | 0.433 | 2.037 | 6.206 |
| Rearrange | no-tail | 6.032 | 0.746 | 5.247 | 7.738 |
| | **Full** | 3.417 | 0.517 | 2.893 | 5.864 |
| Swap Blocks | no-tail | 10.060 | 0.963 | 8.110 | 14.490 |
| | **Full** | 3.486 | 0.548 | 3.127 | 6.314 |
| Pooled | no-tail | 7.510 | 0.851 | 6.491 | 10.751 |
| | **Full** | 3.262 | 0.508 | 2.767 | 6.118 |
| | **Reduction (%)** | **56.6** | **40.4** | **57.4** | **43.1** |

Full is lower on all four diagnostics for every task in Table A4. After count-weighted pooling, the reductions are 56.6%, 40.4%, 57.4%, and 43.1% for position CD, orientation CD, boundary second difference, and overlap RMSE, respectively.

### C.3 Post-Decoding Controls

Linear Continuation applies the execution-side overlap blend without Tail tokens or Tail-aligned initialization, while Temporal Ensemble aggregates actions over time. These controls test whether reducing an output seam can replace Motion Tail's model-side conditioning.

CD and B2 use executed commands; RMSE compares plans. These metrics do not measure accumulated state deviation or fixed-frequency smoothness. Linear Continuation has the lowest seam values but only 0–2% success (Table A5).

**Table A5:** Handoff controls. Success uses 100 episodes per task and method. Boundary means use 20 Put Back and 30 Swap Blocks pairs. Units follow Table A4. Lower is better.

| Method | Succ. | $CD_p$ | $CD_R$ | B2 | $RMSE_3$ |
|---|---|---|---|---|---|
| *Put Back Block* | | | | | |
| no-tail | 0 | 5.900 | 0.842 | 5.930 | 9.660 |
| Linear Cont. | 0 | **2.090** | **0.391** | **1.660** | **5.170** |
| Temporal Ens. | 0 | 3.570 | 0.604 | 3.400 | 5.340 |
| **Full** | **96** | 2.694 | 0.433 | 2.037 | 6.206 |
| *Swap Blocks* | | | | | |
| no-tail | 2 | 10.060 | 0.963 | 8.110 | 14.490 |
| Linear Cont. | 2 | 3.720 | **0.426** | **1.380** | **4.190** |
| Temporal Ens. | 0 | 7.520 | 0.731 | 6.510 | 8.210 |
| **Full** | **74** | **3.486** | 0.548 | 3.127 | 6.314 |

# D Scope and Limitations

**Evaluation regime and statistical scope.** All reported results come from fixed-rate simulation on RMBench and LIBERO, so the boundary diagnostics describe the commanded stream rather than physical trajectories under actuation limits. We do not evaluate on hardware or vary the control rate, and the reported margins should not be read as predictions for asynchronous or real-time deployment. Each internally trained condition uses a single random seed, so we report no seed variability or significance tests. The reported success rates and boundary means should therefore be interpreted as point estimates under this protocol.

**Interpretation of interventions and diagnostics.** Each ablation withdraws a pathway from training and deployment together, so it measures dependence of the assembled system on that pathway rather than isolating an internal mechanism. Likewise, Table A4 compares policies that diverge within matched scenes; it establishes an association between lower boundary diagnostics and higher success, not causation. The asymmetry between the two state components is consistent with motion continuity protecting the observation sequence, but we do not claim a measured causal chain. All four diagnostics are local to a seam and do not score whole-episode trajectory quality, accumulated state deviation, or fixed-frequency physical smoothness. Success rate remains the criterion for task completion.

**External baseline comparability.** The source-reported comparison rows in the main paper were not rerun under our protocol and may differ in backbone, demonstration count, optimization budget, and success criterion. Only the internally trained conditions share the settings listed above, so the ablation contrasts are matched while the external comparison is not. We use the external rows for positioning and base the mechanism claims on the matched interventions.

**Fixed design choices.** Horizon and prefix lengths, the memory capacity of 16 records, the write-confidence threshold of 0.55, the fixed three-query write interval, and the overlap-consistency weight of 0.2 were each fixed once and reused across all conditions. We did not search these values, so we cannot report sensitivity to them.